EXPERIENCE PRODUCING SIMULATED EVENTS FOR THE DZERO EXPERIMENT ON THE SAM-GRID

G. Garzoglio[#], I. Terekhov, FNAL, Batavia, IL 60510, USA
J. Snow, Langston University, Langston, OK 73050, USA
S. Jain, A. Nishandar, University of Texas at Arlington, Arlington, TX 76019, USA

Abstract

Most of the simulated events for the DZero experiment at Fermilab have been historically produced by the "remote" collaborating institutions. One of the principal challenges reported concerns the maintenance of the local software infrastructure, which is generally different from site to site. As the understanding of the distributed computing community over distributively owned and shared resources progresses, the adoption of grid technologies to address the production of Monte Carlo events for high energy physics experiments becomes increasingly interesting. SAM-Grid is a software system developed at Fermilab, which integrates standard grid technologies for job and information management with SAM, the data handling system of the DZero and CDF experiments. During the past few months, this grid system has been tailored for the Monte Carlo production of DZero. Since the initial phase of deployment, this experience has exposed an interesting series of requirements to the SAM-Grid services, the standard middleware, the resources and their management and to the analysis framework of the experiment. As of today, the inefficiency due to the grid infrastructure has been reduced to as little as 1%. In this paper, we present our statistics and the "lessons learned" in running large high energy physics applications on a grid infrastructure.

INTRODUCTION

SAM-Grid is an integrated grid infrastructure for job, data and information handling. Its goal is to enable fully distributed computing for the second run of data taking of the DZero and CDF experiments at Fermilab, Batavia, Illinois. The SAM-Grid project integrates standard grid technologies, such as the Globus Toolkit and Condor-G, for job and information management (JIM) [1, 2] with software developed at Fermilab for data handling, the Sequential Access via Metadata system (SAM) [3, 4].

While the SAM system has been used in production since 1999, the full SAM-Grid infrastructure, which comprises job and information management as well as data handling, has only been deployed for production since January 2004. The system is currently used to produce simulated events for DZero and it is under development to allow data reconstruction for DZero and Monte Carlo production for CDF. As of today, the system has produced about 2 million events, equivalent to about 10 years of computation on a typical GHz CPU.

During the initial phase of deployment, between January and March 2004, the inefficiency in event production[*] due to the grid infrastructure has been reduced from 40% to 1-5%. This paper describes the problems that we have faced during the deployment phase and subsequent operations, and it explains the solutions adopted to decrease the production inefficiency.

The paper is organized in two main sections. First, we describe the SAM-Grid deployment model, in order to stress the similarity to other grid infrastructures as far as software and hardware layout is concerned. Second, we list the problems encountered during the deployment and the solutions adopted. The list is organized in three broad categories: system or cluster problems, gateway or grid/fabric interface problems, and grid services problems.

THE SAM-GRID DEPLOYMENT

The services of a grid architecture can be generally organized in two distinct layers: the grid layer, which encompasses those services that are global in nature, and the fabric layer, which includes services whose scope is restricted to individual sites. The two layers interact via an interface, which adapts the generic directives of the grid services to the peculiarity of the configuration of the fabric at the site.

Figure 1 shows the division in grid and fabric services for the SAM-Grid architecture. The SAM-Grid grid-level services include the resource selection service, the global data handling service, such as metadata and replica catalogue, and the submission services, which are responsible for maintaining the queue of grid jobs and for interacting with the remote resources at the sites. The fabric services include the local data handling and storage services, the local monitoring, and the local job scheduler. The most popular interface between the two layers is defined by the Globus Resource Allocation and Management (GRAM) protocol [5]. The Globus Toolkit distributes implementations of different interfaces for various batch systems. These interfaces are called job-managers and have become the *de facto* standard. As we argue in the later section, these job-managers are not sufficient for a complex grid infrastructure. For this reason, the SAM-Grid has developed its own job-managers, adhering to the GRAM protocol.

[#] garzoglio@fnal.gov

[*] The inefficiency is defined as 1 – (events produced / events requested)

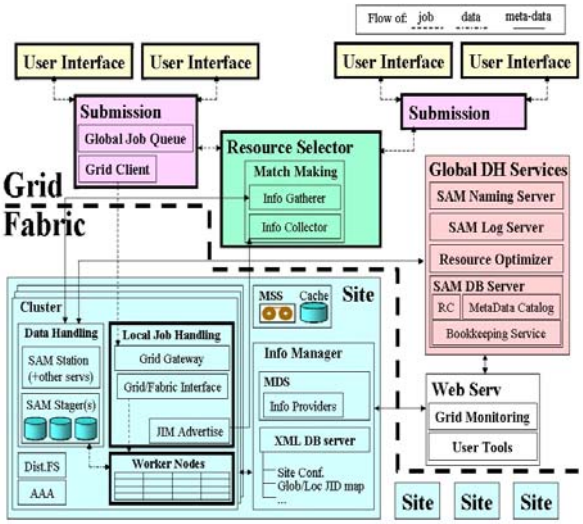

Figure 1: diagram of the SAM-Grid architecture organized in grid and fabric services. The grid services are global in nature, while the fabric services are limited to the scope of a single site

The deployment phase consisted in installing and configuring software at the collaborating sites so that they could accept jobs from the SAM-Grid grid services. The sites generally offered a gateway machine and administrative support in order to install the standard middleware from the Virtual Data Toolkit (VDT) distribution [6], the SAM-Grid grid/fabric interface, and the client software for the fabric services. The fabric services were either deployed on the gateway node itself or on different machines nearby. It should be noted that the SAM-Grid does not require any preinstalled software or running daemons at the worker nodes of the cluster.

The SAM-Grid is currently deployed at a dozen sites in the US and Europe, half of which are stable enough to allow production quality job execution. Because the software infrastructure at each site is uniform and adapts to the configuration of the fabric, the maintenance work necessary to run production consists of a single grid administrator with a contact person at each site, in the seldom case where privileged access is needed. This is an improvement on the pre-grid model, where every site needed a person responsible for maintaining the local production scripts and for submitting the jobs locally. In the SAM-Grid model a single user can submit from his client machine to any collaborating site.

THE LESSONS LEARNED

During the deployment phase and subsequent operations, we have encountered a variety of problems for which we present solutions. We organize the problems in three major categories, depending on the location of their occurrence:

- at the cluster: generally stemming from administrative problems with the system
- at the gateway: in the grid/fabric interface
- at the grid services: typically problems in the access to the grid services by the fabric.

Cluster problems

Worker nodes synchronization: grid infrastructures rely on strong authentication mechanisms to grant access to resources. Security tokens are time stamped and their validity is checked against the machine clock. In our experience, maintaining the synchronization within minutes of absolute time is generally enough, since that is the minimum time between when the token is created and when it is used at the collaborating site, considering the typical latencies of a grid system. Various tools are available to system administrators to synchronize the machines clocks, including NTP [7].

Failure in polling the status of a job from the local batch system: the SAM-Grid was initially interfaced to three different batch systems: PBS, BQS, and Condor. After submitting on the order of hundreds of jobs, the SAM-Grid periodically polls their status. In our experience, all of these batch systems, especially when under stress, have failed to report the status of the local jobs, either because the polling request timed out (PBS, Condor) or because the batch system temporarily couldn't find the job in the queue (BQS). It should be noted that this transient condition would not disrupt the activity of an interactive user. To the contrary, it causes the grid to consider the job terminated, thus creating a resource leak. Our attempts to aggregate polling requests, in order to diminish the stress to the batch system, only mitigated the problem. We have therefore written a level of abstraction on top of the batch systems, with the purpose of increasing the reliability of the interaction with them. We refer to this layer as "idealizer", as it idealizes the behaviour of the underlying batch system. We found this technique of fundamental importance to increase the stability of grid operations.

The "Black Hole" effect: even if a single worker node of a cluster has configuration problems that cause the jobs to crash, all the jobs in the queue end up crashing. If the batch system is busy processing long jobs, in fact, the failing node is the only one with a fast turn around and the scheduler will keep sending jobs to it. Using the batch system "idealizer", we have designed ways to statistically discourage submission to suspicious nodes that process long jobs too quickly.

The worker nodes may need to know their domain name: the domain name is a convenient way to express global policies. In the case of the SAM-Grid, the infrastructure selects the "best" file transfer protocol according to a map that includes the domain name. Worker nodes that were not configured to know their domain name could not use the protocol selection mechanism. It is trivial, on the administrative level, to let the worker node know their domain.

Running gridftp transfers between the head and the worker nodes in a private network requires special configuration: the standard gridftp software, which is distributed by the Globus Toolkit, works in "active" mode

only. This means that a client that initiates a transfer from a worker node is responsible for opening the data port. If the server at the head node does not have an interface to the private network, it may not be able to connect to it. The problem appears with the Network Address Translation (NAT) machine failing to translate the worker node address requested by the server. We believe that this problem is related to the implementation of the server that we are using, which come from the Globus Toolkit v2.4.3. Even if it may be possible to solve this problem by changing the NAT configuration, the administrators of our collaborating sites have always opted to give the head node an interface to the private network.

Plan Operating System upgrades with the system administrators or be resilient to the changes: in the SAM-Grid, resources advertise the operating system of the local cluster. Jobs that require a special version of an operating system can require it in the job description. The resource selection mechanism is then responsible to honour the extra requirement. Unplanned operating system upgrades at a site have disrupted SAM-Grid operations at that site in the past.

Study the local policies: lack of understanding of the local policies or badly configured policies result in jobs failing or being delayed. Below are a few examples of how local policies have caused problems to SAM-Grid operations.

- Jobs have failed because we selected as default a batch system queue with a CPU limit too short with respect to the typical length of the jobs. A "good" default for interactive job submission is not necessarily good for grid jobs. The local user community, in fact, may have job requirements that are different from the ones of the grid users.
- We experienced long delays because the maximum number of file transfers allowed by the data handling system was unreasonably low.
- On a condor system, some jobs could never finish. The typical grid jobs were expected to run for about half a day. Because of local resource usage and user priorities, this translated in a very high probability of the grid jobs being pre-empted. We had to allow only short jobs at that site.

Gateway problems

We have found that the standard grid/fabric interfaces, provided by the Globus Toolkit in the form of job-managers, were not sufficient to run production-quality jobs on the SAM-Grid [8]. The standard interfaces, in fact, lack in the following areas:

- **Flexibility**: they interface only to "standard" batch system configurations. None of our initial sites was compliant to the Globus job-managers "standards". For example, as part of a special agreement, the University of Wisconsin at Madison runs some of the DZero jobs on their condor cluster without pre-emption. The intention to take advantage of this local policy must be expressed at the time of local job submission. The submission command is specific and cannot be expressed using the standard job-managers. Another example is the special option used at the IN2P3 computing centre in Lyon, France, to inform the scheduler that a job plans to access data via HPSS, the local mass storage system. In case of HPSS downtime, the batch system can schedule those jobs specially, avoiding crashes due to denial of access to the data. This option is also site specific and cannot be part of the standard job-managers. In general, the job-managers do not provide a way to customize the interface to the local scheduler.
- **Scalability**: the Globus Toolkit instantiates a process at the gateway machine for every grid job entering the site. On the average commodity machine this limits the number of grid jobs to a few hundreds. Thus, the necessity of aggregating multiple local jobs from a single grid job. This aggregation is not part of the standard job-managers.
- **Comprehensiveness**: the Globus job-managers interface to the local batch systems only. There are a series of other fabric services that in general need notification when a job enters a site. The data handling system could start data pre-staging while the job is idle in the scheduler queue. The monitoring system can observe the status of the job in the queue, while it is not running; this cannot be achieved if the job is the entity responsible for sending monitoring information. Database accesses common to all the batch processes can be aggregated, thus reducing dramatically network traffic.
- **Robustness**: the standard job-managers cannot react to temporary problems when interacting with the local scheduler. We have mentioned in the previous section our experience when polling local job statuses. Typically, this problem results resource leaks.

To overcome these problems, the SAM-Grid has developed a suite of job-managers. The interactions with the local batch system, or rather its "idealizer", are mediated via a layer of abstraction, which we call the "batch adapter". The batch adapter is set up at installation time to reflect the specifics of the configuration of the local batch system. Using this extra layer of indirection we could customize the grid/fabric interface to the batch system of every collaborating site, reflecting the peculiarities of the local policies and hardware/software configuration.

The SAM-Grid job-managers aggregate multiple batch jobs from single grid jobs, thus drastically reducing the scalability problem of the standard job-managers. A grid job is split at the gateway node into multiple local jobs, according to local or virtual organization policies. This aggregation is also convenient for the grid users, who can manage their grid jobs as single entities, irrespectively of their local multiplicity.

Finally, the SAM-Grid job-managers notify relevant fabric services when a job enters the site and aggregates batch job database accesses, in order to overcome the lack of "comprehensiveness" of the standard job managers.

Grid problems

Scalability of semi-central services: it is well known that access to semi-central services represent a single point of failure in distributed architectures. Nevertheless, grid infrastructures need to be able to cope with the shortcomings of semi-central services, as integration is often needed in order to achieve production-quality service.

In the case of the SAM-Grid, clients access the metadata catalogue via a semi-central server. As dozens of processes at the worker nodes try to access the service almost at the same time, most network connections get refused. This effect was responsible for the failure of about 30% of the jobs. The problem was virtually eliminated taking 3 actions:

1. Streamline the communication with the server, in order to reduce the connection time
2. Aggregate the communication where information overlap among processes existed: the information was gathered once from the gateway node and distributed to the processes
3. Introduce retrial with exponential back off in case of failure

It should be noted that in order to implement 2 above we have extended the grid/fabric interface discussed in the previous section. The retrials, instead, have been inserted directly in the client code, making the change in principle transparent to the running applications. We believe that both steps can apply directly to other grid infrastructures similar to the SAM-Grid. On the other hand, streamlining the communication to the database is a step specific to the typical queries used and we believe it is harder to generalize.

Firewall configuration: maintaining a consistent functional configuration of the firewalls of the collaborating institutions in the whole grid is a challenge. System administrators generally are willing to open ports in the firewall to specific nodes at the time of the installation. As the grid grows, new nodes should be granted access through the firewall of each institution. Realistically though, the reaction of the system administrators for this types of requests is generally slow and the update of site policies, such as network access, are tough to negotiate.

The SAM-Grid has faced this problem mainly for data transfers. New installations are sometimes interested in copying files that are located only at storage elements of older installations. The transfer clients cannot access the servers of the older installation, since they are behind a firewall.

To address the problem, the SAM-Grid has the ability of routing files through a network of data handling servers. The challenge is still understanding the topology of this network and configuring the routing to overcome the firewall limitations. We envision that as new global services become distributed, the problem of routing information will arise in other domains as well. Thus, the ability of delegating proxy servers to access the information should be considered in these architectures as a primary requirement.

ACKNOWLEDGEMENT

We would like to thank all the members of the SAM-Grid team and the DZero and CDF collaborators that have been helping with the SAM-Grid deployment and operations. We also thank the Condor team for their support and the local system administrators.

REFERENCES

[1] I. Terekhov, et al. "Grid Job and Information Management for the FNAL Run II Experiments", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, Ca, USA, March 2003.

[2] G. Garzoglio, et al. "The SAM-GRID project: architecture and plan.", in Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 423 - 425

[3] I. Terekhov et al., "Meta-Computing at D0"; in Nuclear Instruments and Methods in Physics Research, Section A, NIMA14225, vol. 502/2-3 pp 402 – 406

[4] I. Terekhov, et al. "Distributed Data Access and Resource Management in the D0 SAM System", in Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, Aug. 2001

[5] I. Foster and C. Kasselman, "Globus: A Metacomputing Infrastructure Toolkit", International Journal of Supercomputer Applications, 11(2): 115-128, 1997

[6] I. Foster, "Grid Technologies & Applications: Architecture & Achievements", in Proceedings of Computing in High Energy and Nuclear Physics (CHEP01), Beijing, China, Sep. 2001

[7] http://www.ntp.org/

[8] G. Garzoglio, et al. "The SAM-Grid Fabric services", talk at the IX International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT-03), Tsukuba, Japan; to appear in Nuclear Instruments and Methods in Physics Research, Section A